

Principal Component Analysis (PCA)

Aayush Bajaj

July 10, 2026

Contents

1	setup	1
2	variance maximisation	2
2.1	the first component	2
2.2	the rest, by deflation	2
3	the reconstruction view	3
4	computation via the svd	3
5	how many components?	4
6	whitening	4
7	kernel pca, sketched	4
8	from scratch	5
9	see also	6

pca is the linear algebra exam question that escaped into industry. given a cloud of points in $\mathbb{R}^d$, it finds the orthogonal directions along which the cloud spreads the most, and lets you throw away the rest. two apparently different questions — “which directions carry the most variance?” and “which subspace loses the least when i project onto it?” — turn out to have the same answer, and that answer is an eigendecomposition.

*invented by pearson in 1901,
reinvented by hotelling in 1933,
reinvented weekly by graduate
students ever since*

1 setup

data $x_1, \dots, x_n \in \mathbb{R}^d$, assumed **centred**: $\sum_i x_i = 0$ (subtract the mean first; everything below silently requires this). the sample covariance matrix is

$$S = \frac{1}{n} \sum_{i=1}^n x_i x_i^\top \in \mathbb{R}^{d \times d}, \quad (1)$$

symmetric and positive semi-definite, so it has an orthonormal eigenbasis $Sv_j = \lambda_j v_j$ with $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \geq 0$. the variance of the data projected onto a unit vector w is $\text{Var}(w^\top x) = w^\top S w$.

*the $1/n$ vs $1/(n-1)$ choice shifts
eigenvalues by a constant factor
and changes no direction —
pick one and be consistent*

2 variance maximisation

2.1 the first component

find the unit direction of maximal projected variance:

$$\max_{w \in \mathbb{R}^d} w^\top S w \quad \text{subject to} \quad w^\top w = 1. \quad (2)$$

the constraint matters — without it, scale w up and the “variance” grows forever. form the lagrangian

$$\mathcal{L}(w, \lambda) = w^\top S w - \lambda(w^\top w - 1), \quad (3)$$

and set the gradient to zero:

$$\nabla_w \mathcal{L} = 2S w - 2\lambda w = 0 \iff S w = \lambda w. \quad (4)$$

every stationary point is an **eigenvector** of S , and at such a point the objective evaluates to

$$w^\top S w = w^\top (\lambda w) = \lambda. \quad (5)$$

so the achievable variances are exactly the eigenvalues, and the maximiser is the top eigenvector v_1 : the **first principal component**, carrying variance λ_1 (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020).

2.2 the rest, by deflation

the second component solves the same problem with the extra constraint $w \perp v_1$. restricting S to the subspace orthogonal to v_1 leaves a symmetric matrix whose top eigenvector is v_2 , and by induction the k -th component is the k -th eigenvector. collecting the top k as columns of $W_k = [v_1 \cdots v_k]$, the **scores** $z_i = W_k^\top x_i \in \mathbb{R}^k$ are the low-dimensional code, with the tidy properties that the coordinates of z are uncorrelated and $\text{Var}(z_{(j)}) = \lambda_j$.

equivalently: subtract off the rank-one piece and re-run — same answer, worse numerics

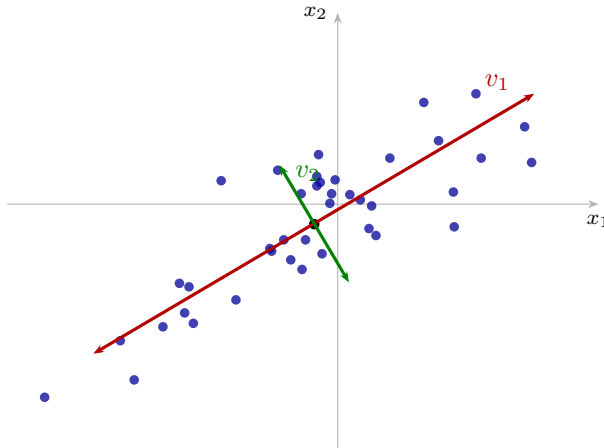


Figure 1: a correlated 2d point cloud with its principal axes drawn at $\pm 2\sqrt{\lambda_j}$ from the mean: v_1 chases the spread, v_2 mops up what is left.

3 the reconstruction view

now the other question: among all k -dimensional subspaces, which one loses the least when we project onto it? for an orthonormal $W \in \mathbb{R}^{d \times k}$ (so $W^\top W = I_k$), the projection of x is $WW^\top x$ and the average squared reconstruction error is

$$\begin{aligned} J(W) &= \frac{1}{n} \sum_{i=1}^n \|x_i - WW^\top x_i\|^2 \\ &= \frac{1}{n} \sum_{i=1}^n (\|x_i\|^2 - x_i^\top WW^\top x_i) \\ &= \text{tr}(S) - \text{tr}(W^\top SW), \end{aligned}$$

using $\|x - WW^\top x\|^2 = x^\top x - x^\top WW^\top x$ (pythagoras: projection and residual are orthogonal). the first term is fixed, so **minimising reconstruction error is exactly maximising the projected variance** $\text{tr}(W^\top SW)$ — the two views coincide. the trace is maximised over orthonormal W by the top- k eigenvectors,¹ giving optimal error

$$J(W_k^*) = \sum_{j=k+1}^d \lambda_j, \quad (6)$$

the **sum of the discarded eigenvalues**. small trailing eigenvalues mean the cloud genuinely lives near a k -flat, and pca finds it (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020).

4 computation via the svd

stack the centred data as rows of $X \in \mathbb{R}^{n \times d}$ and take the (thin) singular value decomposition

$$X = U\Sigma V^\top, \quad U \in \mathbb{R}^{n \times r}, \Sigma = \text{diag}(\sigma_1, \dots, \sigma_r), V \in \mathbb{R}^{d \times r}. \quad (7)$$

then $S = \frac{1}{n} X^\top X = V \frac{\Sigma^2}{n} V^\top$, so

- the **right singular vectors** V are the principal components,
- eigenvalues come from singular values as $\lambda_j = \sigma_j^2/n$,
- the score matrix is $Z = XV = U\Sigma$ — no second pass over the data.

in practice you compute pca via the svd of X , **not** by forming S and calling an eigensolver: forming $X^\top X$ squares the condition number, so half your significant digits evaporate before the eigensolver even starts. when $d \gg n$ (genomics, images) the trick inverts: eigendecompose the $n \times n$ gram matrix $\frac{1}{n} XX^\top$ instead, which shares the nonzero spectrum, then map back — this is also the doorway to kernel pca below.

same reason you solve least squares with qr or svd rather than the normal equations

¹the ky fan theorem: $\max_{W^\top W = I_k} \text{tr}(W^\top SW) = \sum_{j=1}^k \lambda_j$, attained by any orthonormal basis of the top- k eigenspace. the $k = 1$ case is the rayleigh quotient bound used in the lagrangian derivation.

5 how many components?

the eigenvalue spectrum is the budget sheet. define the **proportion of variance explained** by the top k :

$$\text{pve}(k) = \frac{\sum_{j=1}^k \lambda_j}{\sum_{j=1}^d \lambda_j}. \quad (8)$$

- **scree plot**: plot λ_j against j and look for the elbow where signal gives way to noise floor.
- **threshold**: keep the smallest k with $\text{pve}(k) \geq 0.9$ (or whatever your downstream task forgives).
- **scale first**: pca is not scale-invariant. a feature measured in millimetres will dominate one measured in kilometres for no honest reason — standardise features (pca on the correlation matrix) unless their units are genuinely commensurate (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

6 whitening

pca decorrelates; whitening goes one step further and equalises. with $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_k)$, the **pca-whitened** code

$$z_i = \Lambda^{-1/2} W_k^\top x_i \quad (9)$$

has identity covariance: every direction contributes unit variance. useful before algorithms that assume isotropy (nearest neighbours, some optimisers). **zca whitening** $z_i = W \Lambda^{-1/2} W^\top x_i$ (full W , rotated back into the original basis) is the whitening transform closest to the identity — images stay looking like images. caveat: dividing by $\sqrt{\lambda_j}$ for tiny λ_j amplifies noise $1/\sqrt{\lambda_j}$ -fold, so regularise with $\Lambda + \varepsilon I$ or truncate first.

7 kernel pca, sketched

pca can only find flat structure. kernel pca runs the gram-matrix formulation in a feature space $\varphi(x)$ you never compute, touching data only through $k(x, x') = \varphi(x)^\top \varphi(x')$ (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009):

- **centre** the kernel matrix: $\tilde{K} = HKH$ with $H = I - \frac{1}{n} \mathbf{1}\mathbf{1}^\top$ (feature-space centring done implicitly).
- **eigendecompose** $\tilde{K} \alpha_j = n \lambda_j \alpha_j$; normalise so $\lambda_j n \alpha_j^\top \alpha_j = 1$, which makes the implicit feature-space component unit length.
- **project** a point via kernel evaluations only: $z_{(j)}(x) = \sum_{i=1}^n \alpha_{j,i} \tilde{k}(x_i, x)$.

with an rbf kernel this unrolls spirals and separates concentric rings that vanilla pca is congenitally blind to. costs: $O(n^2)$ memory, $O(n^3)$ eigendecomposition, and no cheap inverse map from code space back to inputs (the “pre-image problem”). for visualisation-grade nonlinear reduction the modern defaults are t-sne and umap (McInnes, Leland and Healy, John and Melville, James, 2020), which optimise neigh-

bourhood preservation rather than variance — see also [kernel methods](#) for the general kernel trick, and the [curse of dimensionality](#) for why any of this is necessary.²

8 from scratch

both routes — eigendecomposition of the covariance and svd of the centred data — on a synthetic 5-d dataset with a planted 2-d signal subspace plus isotropic noise:

```

1  import numpy as np
2
3  rng = np.random.default_rng(42)
4
5  # 200 points in 5-d with a planted 2-d signal subspace
6  n, d = 200, 5
7  latent = rng.normal(size=(n, 2)) @ np.diag([3.0, 1.5]) # strong directions
8  basis, _ = np.linalg.qr(rng.normal(size=(d, 2)))        # random orthonormal
9  ↪ 5x2
10 X = latent @ basis.T + 0.3 * rng.normal(size=(n, d))    # + isotropic noise
11
12 def pca_eig(X, k):
13     Xc = X - X.mean(axis=0)
14     S = Xc.T @ Xc / (len(X) - 1) # sample covariance
15     lam, W = np.linalg.eigh(S)    # ascending order
16     lam, W = lam[::-1], W[:, ::-1] # descending
17     return lam[:k], W[:, :k], Xc
18
19 def pca_svd(X, k):
20     Xc = X - X.mean(axis=0)
21     U, s, Vt = np.linalg.svd(Xc, full_matrices=False)
22     lam = s**2 / (len(X) - 1) # singular values -> eigenvalues
23     return lam[:k], Vt[:, :k].T, Xc
24
25 lam_e, W_e, Xc = pca_eig(X, 5)
26 lam_s, W_s, _ = pca_svd(X, 5)
27
28 print("eigenvalues (eig):", np.round(lam_e, 4))
29 print("eigenvalues (svd):", np.round(lam_s, 4))
30 # columns may differ by sign; align before comparing
31 signs = np.sign(np.sum(W_e * W_s, axis=0))
32 print("max |W_eig - W_svd| after sign fix:",
33       f"{np.abs(W_e - W_s * signs).max():.2e}")
34
35 pve = lam_e / lam_e.sum()
36 print("proportion of variance explained:", np.round(pve, 4))
37 print("cumulative:", np.round(np.cumsum(pve), 4))
38
39 # reconstruction error for k = 1..5 equals the sum of discarded eigenvalues
40 for k in range(1, 6):
41     Z = Xc @ W_e[:, :k] # project

```

²the autoencoder connection, precisely: a linear autoencoder $x \mapsto W_2 W_1 x$ with a k -unit bottleneck and squared loss attains its global minimum exactly when $W_2 W_1$ is the projector onto the top- k principal subspace (baldi & hornik 1989) — gradient descent on it rediscovers pca, though without orthogonality or ordering of the individual units. replace the linear maps with deep nonlinear encoders/decoders and you get the modern autoencoder family (Goodfellow, Ian, 2016).

references

Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon (2020). *Mathematics for Machine Learning*, Cambridge University Press. Goodfellow, Ian (2016). *Deep Learning*, MIT Press. Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome (2009). *The Elements of Statistical Learning*, Springer. McInnes, Leland and Healy, John and Melville, James (2020). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*.

```

41 err = np.mean(np.sum((Xc - Z @ W_e[:, :k].T)**2, axis=1))
42 print(f"k={k}: mean sq reconstruction error {err:.4f} "
43       f"sum of discarded eigenvalues {(lam_e[k:].sum() * (n-1)/n):.4f}")

```

```

eigenvalues (eig): [8.9681 1.958  0.1099 0.0868 0.072 ]
eigenvalues (svd): [8.9681 1.958  0.1099 0.0868 0.072 ]
max |W_eig - W_svd| after sign fix: 9.55e-15
proportion of variance explained: [0.8011 0.1749 0.0098 0.0078 0.0064]
cumulative:                       [0.8011 0.976  0.9858 0.9936 1.    ]
k=1: mean sq reconstruction error 2.2156    sum of discarded eigenvalues 2.2156
k=2: mean sq reconstruction error 0.2674    sum of discarded eigenvalues 0.2674
k=3: mean sq reconstruction error 0.1580    sum of discarded eigenvalues 0.1580
k=4: mean sq reconstruction error 0.0716    sum of discarded eigenvalues 0.0716
k=5: mean sq reconstruction error 0.0000    sum of discarded eigenvalues 0.0000

```

everything the theory promised, numerically: the two factorisations agree to machine precision (up to sign — eigenvectors are only defined up to ± 1), the planted 2-d subspace announces itself as two dominant eigenvalues carrying 97.6% of the variance, and the reconstruction error at each k equals the discarded eigenvalue mass exactly (the $(n - 1)/n$ factor in the check just reconciles the two variance conventions).

9 see also

- [k-means clustering](#) — the other half of ch. 14: compress rows instead of columns
- [gaussian mixture models](#) — probabilistic latent structure rather than geometric
- [autoencoders](#) — pca with the linearity constraint deleted
- [kernel methods](#) — the machinery behind kernel pca
- [curse of dimensionality](#) — why high-dimensional clouds need flattening at all